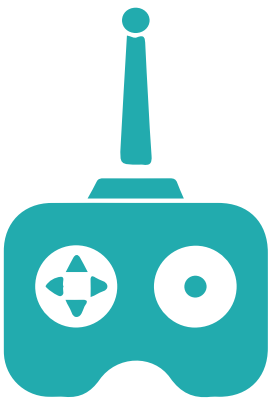
Modul 2

Bug Panic



Impressum

Herausgeber
Landesanstalt für Kommunikation Baden-Württemberg (LFK)
Anstalt des öffentlichen Rechts,
vertreten durch den Präsidenten Dr. Wolfgang Kreißig
Reinsburgstraße 27
70178 Stuttgart

Gefördert durch das Ministerium für Kultus, Jugend und Sport Baden-Württemberg

Autor
Chris Binder

Redaktion
Laura Jaenicke, Landesanstalt für Kommunikation BW (LFK)
Dejan Simonović, Stadtmedienzentrum Stuttgart am LMZ BW

Design und Layout
Jana Falkner

Illustrationen
Ilan Backmann

Stuttgart | September, 2022
2. Auflage
Lizenz CC-BY-SA 4.0

Modul 2

Die Handreichung steht unter https://games-im-unterricht.de/toolkit auch als PDF zur Verfügung.

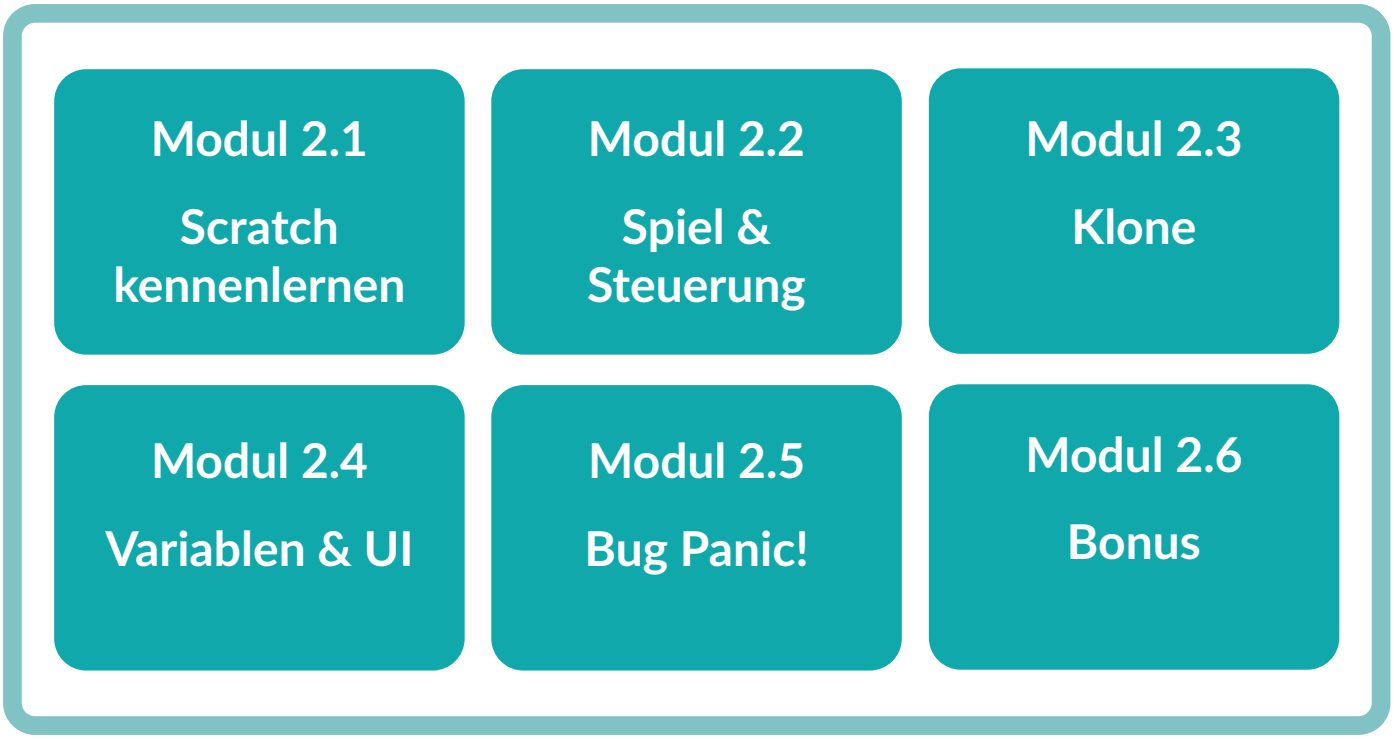
Die Rechte der verwendeten Grafiken und Bilder liegen, soweit nicht anders vermerkt, beim Stadtmedienzentrum Stuttgart.

Das Werk enthält Screenshots aus dem verwendeten Programm Scratch. Diese sind selbst erstellt und werden im Sinne eines Zitats zu Bildungszwecken genutzt.


Chip, wieso sieht hier denn alles so seltsam aus?
Könnte der Bug hier Schuld sein?
Platina! Da drüben ist der echte Übeltäter!

Beschreibung

Modul 2 | Patching

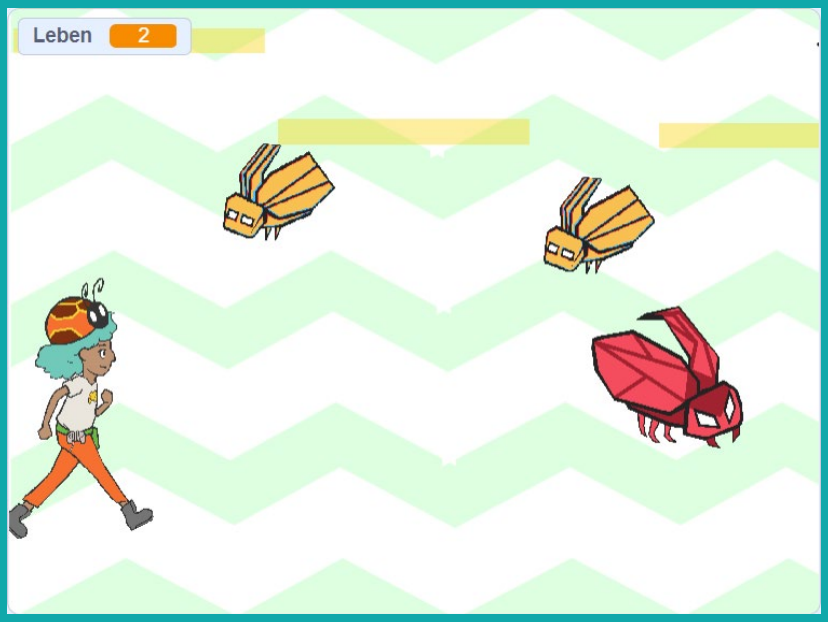


Bug Panic besteht aus fünf Untermodulen und einem Bonusmodul, die jeweils einen Aspekt der Spieleentwicklung beleuchten. So werden durch die Entwicklung eines Fang- und Ausweichspiels zunehmend komplexere Aspekte der Programmierung motivierend vermittelt. Die Untermodule sollten nacheinander bearbeitet werden, da sie eng aufeinander aufbauen.

Mithilfe der Entwicklungsumgebung Scratch programmieren die Schüler:innen ein Ausweichspiel. Scratch läuft im Webbrowser und bietet Codeblöcke zur visuellen Programmierung des Quellcodes. In Scratch können unter anderem Ereignisse, Bedingungen, Schleifen und objektorientierte Programmierung genutzt werden. Dadurch, dass das Spiel jederzeit im Editor ausgeführt und gestoppt werden kann, wird das Prinzip des Debuggings zu einem zentralen und intuitiven Bestandteil des Programmierprozesses. Aufgrund des niedrigschwelligen Einstiegs und der gleichzeitig möglichen Komplexität ist der Umgang mit Scratch für unterschiedlichste Gruppen motivierend.

Nach einem Einstieg in die Grundlagen erstellen die Jugendlichen die Spielumgebung und implementieren die Steuerung des Charakters. Sie erzeugen Klone eines Prototyp-Objekts, die als Gegner dienen. Über Variablen werden dann Spielfortschritt und Zustand des Spielcharakters abgefragt. Im letzten Untermodul schließen die Schüler:innen das Projekt mit Gewonnen-/Game-Over-Screens ab und testen das fertige Spiel.

Ein Bonus-Untermodul bietet kurze Anregungen für eine zusätzliche Einheit, in der Sounds und eigens gestaltete Grafiken eingefügt werden können.



Scratch-Spiel "Bug Panic"

Das vorliegende Modul 2 ist auf den Bildungsplan Baden-Württemberg für den Aufbaukurs Informatik in Klasse 7 ausgerichtet. Dabei werden aus den prozessbezogenen Kompetenzen Strukturieren und Vernetzen als auch Modellieren und Implementieren abgedeckt. Darüber hinaus deckt das Modul alle Kompetenzen des inhaltsbezogenen Bereichs Algorithmen ab (siehe Einleitung, S. 6&7).

Einbindung in die Story

Platina und Chip verfolgen gemeinsam einen Computerfehler, den Protobug. Dieser erzeugt immer wieder neue bösartige Fehler, die Bad Bugs. Das Ziel ist es, in Scratch ein Spiel so zu programmieren, dass Platina und Chip den Bad Bugs ausweichen können und nach einer bestimmten Zeit den Protobug einfangen.

Rahmenbedingungen

Zeit

5 Untermodule und ein Bonusmodul à 90 Minuten

Das Modul besteht aus fünf Untermodulen und einem Bonusmodul, welche jeweils auf 90 min ausgelegt sind.

Raum

Präsenz- oder Fernunterricht

Das Modul kann sowohl im Präzenzunterricht, als auch im Fernunterricht durchgeführt werden.

Technik und Materialien

Geräte : PC/Tablet

Pro Schülerin oder Schüler ein Tablet oder PC, auf dem Scratch ausgeführt werden kann (https://scratch.mit.edu/info/faq/#about-scratch).

Online
Lehraccount auf scratch.mit.edu
Bug-Panic-Projektdatei (https://scratch.mit.edu/studios/29218931/)

Offline
Scratch 3 (https://scratch.mit.edu/download)

Vorbereitung

Für dieses Modul steht mit einem Scratch-Studio eine Kollektion bereit, in der die benötigten Scratch-Projekte online angeschaut, verändert und für den eigenen Gebrauch kopiert werden können:

https://scratch.mit.edu/studios/29218931/

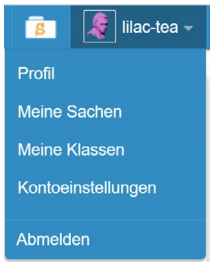
Um Schüler:innen besser online verwalten zu können, empfehlen wir, einen Lehraccount anzulegen. Da der Account erst vom Scratch-Team geprüft wird, sollten Sie diesen vor Beginn des Projekts anlegen. Anmeldung und Infos finden sie hier:

https://scratch.mit.edu/educators

Mit einem Lehraccount können Sie Ihre Klassen selbst verwalten und dort Accounts für Schüler:innen selbst anlegen.
Die FAQ-Seite für Lehraccounts bietet einen Überblick und ein (englisches) Einführungsvideo, das alle Funktionen des Lehraccounts bündig erklärt:

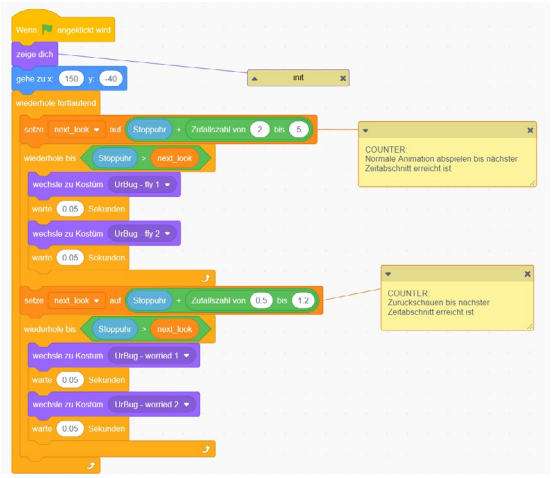
https://scratch.mit.edu/educators/faq

Nutzerfunktionen



Das diesem Modul beigefügte Handbuch bietet eine Einführung in die Benutzeroberfläche, Funktionen und Blöcke von Scratch. Hier können die Schülerinnen und Schüler auch ihre Anmeldedaten vermerken und Notizen aufschreiben.
Machen Sie sich als erstes mit dem Handbuch vertraut, das die Benutzeroberfläche und die grundlegenden Funktionen erklärt.

Benutzeroberfläche



Schauen Sie sich das fertige Spiel Bug Panic an und vollziehen sie den kommentierten Code nach:

https://scratch.mit.edu/projects/473817357

Mit der Schaltfläche 

können Sie den Programmcode aufrufen.

Dieser ist an vielen Stellen durch zusätzliche gelbe Notizzettel kommentiert.

Durchführung Modul 2 Bug Panic

2.1 Scratch kennenlernen

Die Schüler:innen lernen die Benutzeroberfläche von Scratch kennen und entwickeln angeleitet ihre erste Animation. Sie passen ihr Programm dann an und lernen so über die Grundlagen der Programmierung und in welcher Reihenfolge die Anweisungen ausgeführt werden.

Das erste Untermodul kann durch eine Einführung in die Story (z.B. Was würdet Ihr an Platinas und Chips Stelle tun?) und durch eine Abfrage von EDV-/Spielkompetenzen eingeleitet werden.

Die grundlegenden Funktionen von Scratch sollten mit den Schüler:innen gemeinsam durchgegangen werden. Auf einigen Geräten ist die Oberfläche auf Englisch eingestellt, diese kann über das Weltsymbol umgestellt werden. Dabei wird bestehender Programmcode nicht gelöscht, sondern übersetzt. Die Schüler:innen sollten eigene Projekte eindeutig betiteln und regelmäßig abspeichern. Sobald ein Projekt durch Veröffentlichen veröffentlicht wurde, kann es von anderen Schüler:innen und durch den Lehraccount eingesehen werden. Dies ist zum Nachvollziehen des Programmcodes besonders praktisch.

Das erste Programm, in dem Platina Chip begrüßt und sich zu diesem hinbewegt, sollte erklärt werden, um den Einstieg in die Syntax von Scratch zu erleichtern. Dabei können die Schüler:innen durch Remixen (remixen) des Projekts Bug Panic Intro mit vorgefertigten Figuren arbeiten. Am besten kopieren Sie das Projekt vor Modulbeginn in ein Studio, das Sie für Ihre Klasse erstellen.

Bug Panic Intro

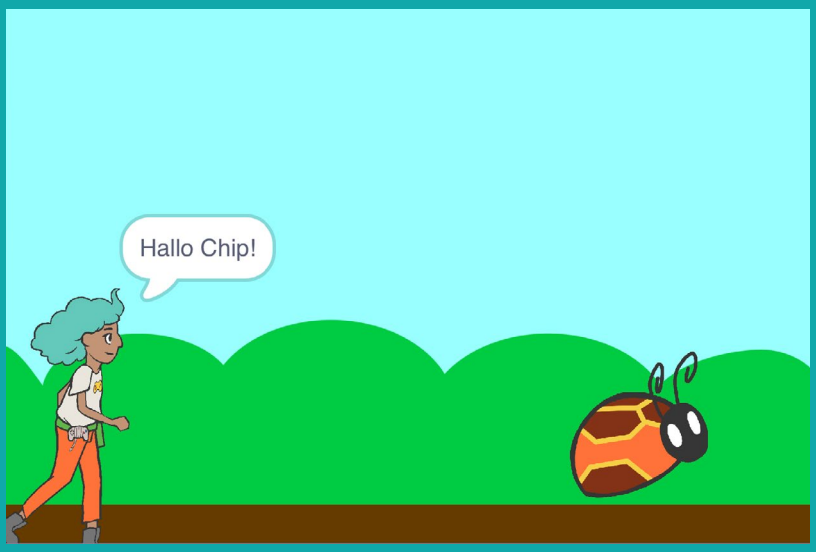
https://scratch.mit.edu/projects/474082492/

Danach können die Schüler:innen Werte im Programmcode verändern und die Resultate in eigene Worte fassen oder ausgehend von Fragen wie “Wie würdet Ihr die Geschwindigkeit von Chip verändern?” das Spiel anpassen.

In einem Abschlussgespräch wird den Schüler:innen die Möglichkeit gegeben, sich zu ihren Erfahrungen zu äußern und Eindrücke zu schildern. Hier kann auch ein Ausblick gegeben werden, indem auf weitere Schritte und Ansätze hingewiesen wird.

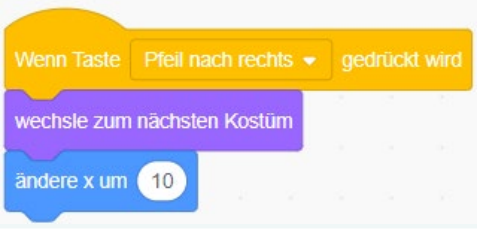
Verlaufsplan | 2.1 Scratch kennenlernen

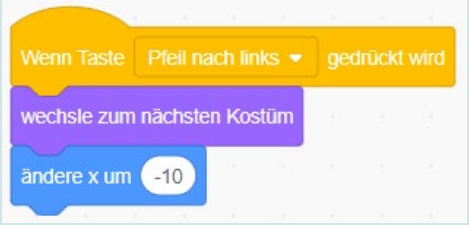
Dauer	Phase und Unterrichts-form	Inhalt	Material und Technik
5-10 min	Einstieg, Unterrichts-gespräch	Warm-up, z.B. Programmier- und Spielkompetenzen abfragen	
10 min	Einführung	Gemeinsam in Scratch einloggen und Signup abschließen	Lehraccount
10-15 min	Einführung	Benutzeroberfläche erklären • Spracheinstellungen • Titel, Speichern • Meine Sachen • (Klasse)	Beamer
20 min	Erarbeitung, Unterrichts-gespräch und Einzel-/Part-nerarbeit	Gemeinsam erstes Programm ent-wickeln: • Klicksteuerung Platina • Grüne Flagge: Chip wartet, bis Platina bei ihm ist. • Einführung Werte mit X-Y-Position • Zustände: Grüne Flagge initialisiert Spiel	Scratch-Projekt „Bug Panic Intro“
20 min	Problema-tisierung, Einzel-/Part-nerarbeit	SuS sollen vorhandenes Programm anpassen: • Was sagt Platina? • Wie schnell ist Platina? • Wie schnell ist Chip?	
10-15 min	Unterrichts-gespräch/ Präsentation	SuS präsentieren Ergebnisse und Lösungswege	



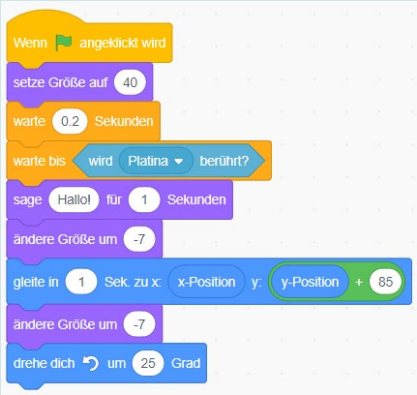
Screenshot Scratch

Programm Platina





Programm Chip



Anknüpfpunkte Informatik | 2.1 Scratch kennenlernen

Informatik	Modul 2.1
Bedingungen	Auslöser für eine Handlung, z.B. berührt Platina Chip?
Werte	Namen und X- und Y-Positionen der Figuren
Zustände	Start- und Endzustand des Programms, Anfangszustand am Anfang des Spiels (Initialisierung)
Timer	Genutzt für Warten-Anweisung, gleite in XX Sekunden…
Refactoring	Code schlanker gestalten und Anweisungen vereinfachen

2.2 Spielumgebung und Steuerung

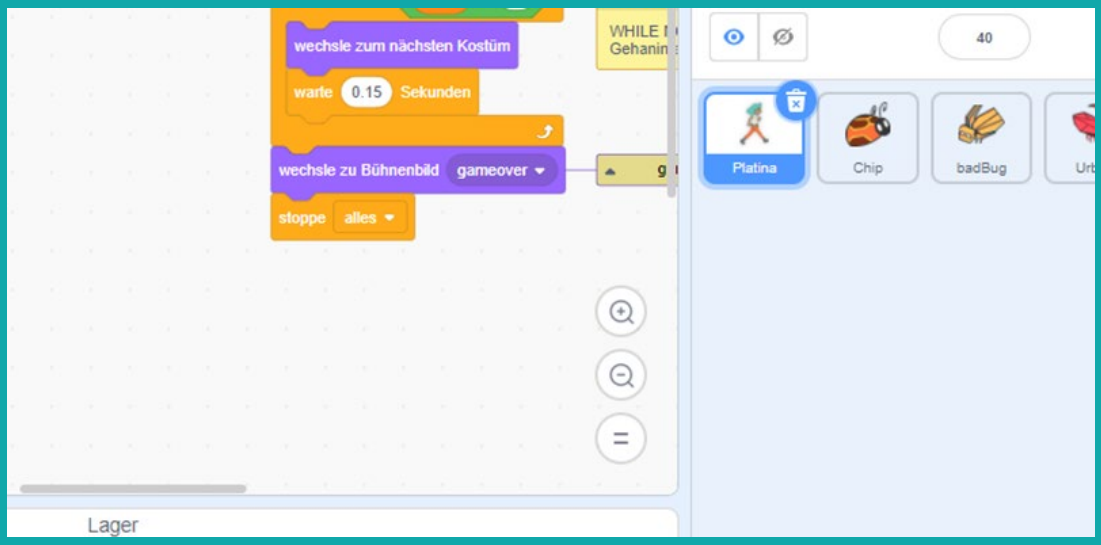
Die Schüler:innen importieren Grafiken aus einem Scratch-Projekt oder laden diese von einem Computer/Tablet hoch. Sie lernen den Kostümeditor kennen und passen Grafiken grundlegend an die eigenen Bedürfnisse an. Zudem wird eine Schleife für die Steuerung des Charakters Platina programmiert.

Das Projekt Bug Panic – Materialien enthält alle benötigten Grafiken, nicht aber den Programmcode. Das Projekt ist unter folgender URL zu finden und kann mit Klick auf Remixen (remixen) in den eigenen Lehraccount kopiert werden:

https://scratch.mit.edu/projects/488800073/

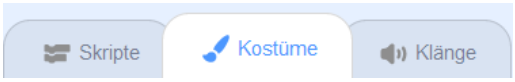
Die Figuren können, sofern die Schüler:innen angemeldet sind, in das Lager am unteren Bildschirmbereich gezogen werden. Von hier können sie in beliebigen Projekten weiterverwendet werden.

Platina Sprite



Screenshot Scratch Codeblöcke

Scratch Menu



Danach können die Schüler:innen den Kostümeditor verwenden, um die vorhandenen Grafiken leicht anzupassen. Der Kostümeditor ist über die Schaltfläche oberhalb der verfügbaren Blöcken erreichbar.

Um eine Heldin zu haben, die steuerbar ist, sollten die Schüler:innen eigenständig eine Steuerung entwickeln. Neben dem Referenzbeispiel mit einer While-Schleife („wiederhole fortlaufend“ & „warte bis“, siehe Modul2 - Handbuch) gibt es weitere Möglichkeiten, um Platina zu bewegen. Die Schüler:innen sollten ihren Code auch testen, um deren Funktionalität zu überprüfen.

Im Toolkit-Koffer sind Gamecontroller enthalten, die benutzt werden können, um das Spiel zu testen. Zum Konfigurieren der Steuerung liegt dem Toolkit eine Erklärung bei, wie Sie die Tastenanschläge des Gamecontrollers anpassen können.

Tipp: Codeblöcke lassen sich über das Kontextmenü (Rechtsklick) duplizieren.

Das Prinzip verschiedener Figuren, die jeweils ihre eigenen Codeblöcke ausführen, sollte aus dem ersten Modul bekannt sein und kann aufgefrischt werden, in dem Chip immer über Platina platziert wird. Mit einer weiteren Schleife kann der Protobug dazu gebracht werden, immer auf der Stelle zu rennen.

Verlaufsplan | 2.2 Spielumgebung und Steuerung

Dauer	Phase und Unterrichts-form	Inhalt	Material und Technik
5-10 min	Einstieg, Einzel-/Partner-arbeit	Grafiken importieren	Dateiset / Scratchdatei „Bug Panic - Materialien“
10-15 min	Erarbeitung I, Einzel-/Partnerarbeit	Intro Kostümeditor importieren, transformieren, einfärben, kopieren+einfügen, spiegeln Differenzierung: Text	
20-25 min	ErarbeitungII, Einzel-/Partnerarbeit	Steuerung von Platina programmieren (While-Schleife); Unterschied zu „Wenn Taste gedrückt wird“	
5 min	Gruppen-arbeit/ Austausch	SuS wählen gut funktionierende Codestelle und schauen sich dann die Codes der anderen SuS an	
20 min	Erarbeitung III, Einzel-/ Partnerarbeit	Chip immer bei Platina platzieren; Urbug-Lauf-Animation	
10-15 min	Sicherung/ Reflexion	SuS fassen gut gelungene Funktionen in eigene Worte und reflektieren über Erfahrungen beim Implementieren der Funktionen	

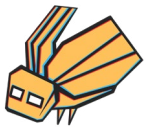
Anknüpfpunkte Informatik | 2.2 Spielumgebung und Steuerung

Informatik	Modul 2.2
Assets	Materialien wie Texturen oder Sounds (kein Code), die für das Spiel notwendig sind
Liste(-n)	Datentyp, der mehrere Werte listenartig speichert, z.B. Kostüme oder Hintergründe
While-Schleife	Um Anweisungen zu wiederholen, während ein bestimmter Programmzustand besteht

2.3 Klone

Aus einem Urobjekt werden Klone erstellt, die nach einer zufälligen Zeit erzeugt werden. Diese werden bewegt und wieder gelöscht, wenn sie nicht mehr benötigt werden. Die Schüler:innen programmieren eine Bedingung für die Kollision zwischen den Bad Bugs und Platina.
Zu Anfang sollte das Prinzip der Klone erläutert werden. Hierbei ist zu beachten, dass Klone alle Eigenschaften des Urobjekts erben. Wenn das Urobjekt versteckt ist, sind die Klone zunächst auch versteckt. Mit der Anweisung „Wenn ich als Klon entstehe“ können alle Aufgaben definiert werden, die die Bad Bugs ausmachen: Diese sollen zufällig erzeugt werden und von rechts nach links durch das Bild krabbeln. Die Klone müssen ebenfalls Bedingungen für den Fall enthalten, dass sie den Rand erreichen („lösche diesen Klon“) oder Platina berühren. Das Handbuch enthält praktische Befehlsgrundlagen für diese Anweisungen.

Bad Bug



Verlaufsplan | 2.3 Klone

Dauer	Phase und Unterrichtsform	Inhalt	Material und Technik
10 min	Einstieg	Demonstration Klone	
20 min	Erarbeitung I, Einzel-/Partnerarbeit	Bad Bug als verstecktes Objekt programmieren, das böse Bugs erzeugt.	Scratch-Projekte der SuS
10 min	Reflexion, Unterrichtsgespräch	Besprechung und Bewertung der Ergebnisse im Plenum, z.B. durch Hervorheben interessanter Ansätze beim Programmieren	
30 min	Einzel-/Partnerarbeit	SuS Klone anhand von „Wenn ich als Klon entstehe“ programmieren lassen. Ziele: • Klone bewegen lassen • Klone links am Rand löschen • Klone auf Berührung mit Platina prüfen lassen	
10 min	Sicherung/ Reflexion	SuS präsentieren Ergebnisse und reflektieren über Lösungsansätze/-wege	

Anknüpfpunkte Informatik | 2.3 Klone

Informatik	Modul 2.3
Objekte	Elemente einer Klasse, in diesem Fall die Bad Bugs
Vererbung	Bad Bugs bekommen alle Eigenschaften des ursprünglichen Bugs
Prototyp	Protobug

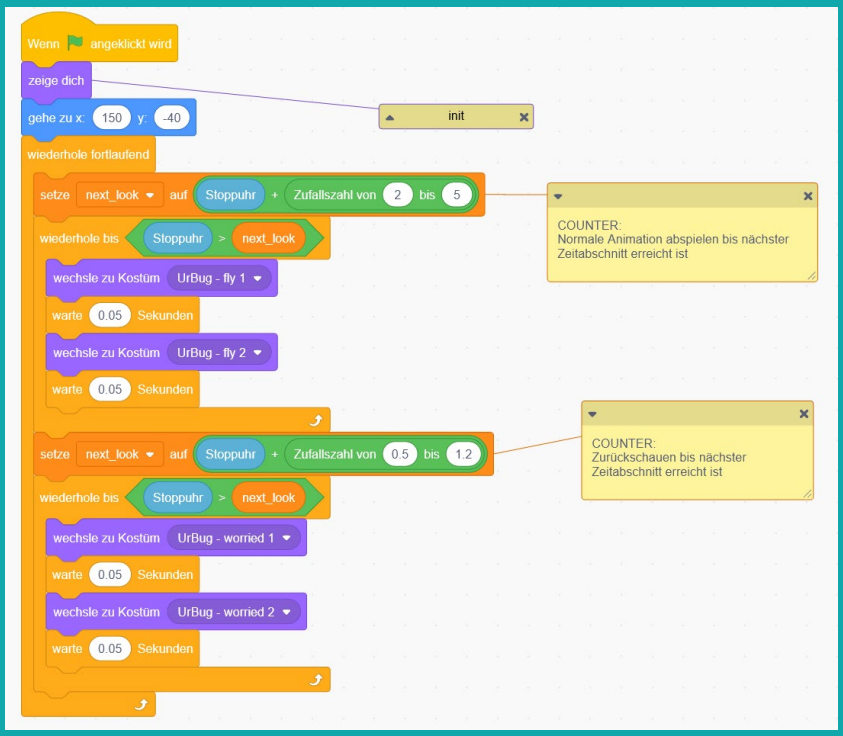
2.4 Variablen und UI

Auf der Zielgeraden zum fertigen Spiel werden Hintergründe hinzugefügt, zufällige Zeitwerte für das Erscheinen der Bad Bugs definiert und ein Game-Over-Screen eingefügt, der von den Schüler:innen programmiert wird.

Das nahezu fertige Spiel kann durch die Schüler:innen um eigene Hintergründe für “Game-Over” und “Win” erweitert werden.

Um Abwechslung zu erzeugen, kann der besorgte Zustand des Protobugs eingefügt werden, der in zufälligen Abständen aktiviert und deaktiviert wird. Auch Bad Bugs können mit der Zufallszahl-Funktion in unvorhersehbaren Abständen erzeugt werden.

Zuletzt könnten die Schüler:innen das Spiel testen.



Screenshot Scratch Codeblöcke

Verlaufsplan | 2.4 Variablen und UI

Dauer	Phase und Unterrichts-form	Inhalt	Material und Technik
10 min	Einstieg, Unterrichts-gespräch	Über Werte hin zu Variablen führen, Funktionalität zeigen	Evtl. Tafel/White-board
20 min	Erarbeitung I, Einzel-/Part-nerarbeit	SuS verändern „Leben“-Variable und schauen, wann diese geändert werden muss. • bei 1 Leben Chip verstecken • bei 0 Leben alles stoppen • bei Initialisierung auf 2 Leben einstellen	
10 min	Partnerarbeit	SuS tauschen Partner, diese testen jeweils den Code anderer SuS	
25-30 min	Erarbeitung II, Einzel-/ Partnerarbeit	SuS bauen Lebensanzeige ein, die mit der Variable „Leben“ korres-pondiert	
10 min	Sicherung/ Reflexion	SuS stellen eigenen Code vor und erläutern Ablauf des Programms	

Anknüpfpunkte Informatik | 2.4 Variablen und UI

Informatik	Modul 2.4
Variable	Eine Variable ist wie die Pronomen „Er/Sie/Es“. Sie verweist immer auf genau ein Objekt. Wenn man den Namen einer Variablen eingibt, bekommt man das Objekt.

2.5 Bug Panic!

Mit Hilfe von Variablen legen die Schüler:innen den Gesundheitszustand von Platina fest und fragen diesen ab. Die Leben werden visuell dargestellt.

Den Schüler:innen sollte das Prinzip der Variablen vermittelt werden, beispielsweise anhand des Vergleichs mit Pronomen, die auf eine bestimmte Person verweisen und einzigartig sein müssen, um nicht zu Ambivalenzen zu führen.

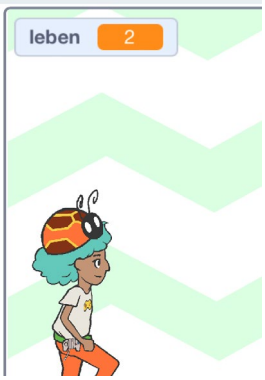
Mit Hilfe der Funktion

im Blockbereich 

kann eine geeignete Variable, z.B. mit dem Namen „Leben“ erzeugt werden.

Die Schüler:innen können den Wert der Variablen dann anpassen, wenn ein Bad Bug Platina berührt. Die Variablen sollten bei jedem Spielstart wieder auf den Ursprungswert eingestellt werden (Initialisierung).
Als Lebensanzeige kann Chip als eigenständige Figur fungieren, die nur sichtbar ist, wenn Platina volle Gesundheit hat.

Lebensanzeige in Scratch



Verlaufsplan | 2.5 Bug Panic!

Dauer	Phase und Unterrichtsform	Inhalt	Material und Technik
10 min	Einführung	Intro Bühnenbilder	Lehraccount und Beamer/Präsentationsgerät
25 – 30 min	Erarbeitung I, Einzel-/Partnerarbeit	Game-Over-Bedingungen festlegen, triggern	Scratch-Projekte der SuS
10 min	Unterrichtsgespräch	Diskussion über Zufallszahlen und deren Nutzung im Code	
15-20 min	Erarbeitung II, Einzel-/Partnerarbeit	Zufällige Werte für zeitbasierte Events nutzen	
10 min	Sicherung, Einzel-/Partnerarbeit	SuS testen die Spiele und geben sich gegenseitig Rückmeldung	Gamecontroller aus dem Toolkit-Koffer

Anknüpfpunkte Informatik | 2.5 Bug Panic!

Informatik	Modul 2
Zufallszahlen	Zufallszahlen nutzen und sie für Programmlogik nutzbar machen • Exkurs tatsächliche Zufallszahlen
Counter	Sekunden und Millisekunden seit Spielstart • Exkurs Variablentypen bzw. mögliche Zahlenformate beim Programmieren

2.6 Bonus

Das Bonus-Untermodul bietet sich für umfangreichere Projekte oder als Differenzierung an. Hier kann das Spiel um eigene Sounds erweitert werden, die in das Spiel geladen und programmiert werden. Zudem können selbst gezeichnete Grafiken eingescannt und als Spielelemente verwendet werden. Die Schüler:innen erlernen so das Asset-Management und die Unabhängigkeit zwischen Grafik/Sound und Funktionalität.
Die Schüler:innen können hier aus der Scratch-Soundbibliothek Sounds auswählen oder eigene Sounds, etwa am Tablet, aufnehmen. Auch können eigene Grafiken mit Hilfe des Kostümeditors gestaltet werden.

Als Differenzierungsaufgabe für besonders versierte Programmierer:innen bietet sich die Möglichkeit, ein Menü zu gestalten, das vor dem eigentlichen Spiel erscheint. Außerdem können Schwierigkeitsgrade implementiert werden, indem zum Beispiel die Geschwindigkeit der Bad Bugs über eine Variable angepasst wird.

Verlaufsplan | 2.6 Bonus

Dauer	Phase und Unterrichts-form	Inhalt	Material und Technik
20 min	Erarbeitung I, Einzel-/Part-nerarbeit	Sounds aufnehmen/auswählen	Scratch-Projekte der SuS
25 min	Erarbeitung II, Einzel-/ Partnerarbeit	Eigene Charaktere zeichnen/ge-stalten und scannen/abfotografie-ren	
20 min	Erarbeitung III, Einzel-/ Partnerarbeit	Menü gestalten	
25 min	Erarbeitung IV, Einzel-/ Partnerarbeit	Schwierigkeitsgrade implementie-ren	
10 min	Sicherung	SuS stellen fertige Sounds und Charaktere vor	

Anknüpfpunkte Informatik | 2.6 Bonus

Informatik	Modul 2.6
Dateiformate	z.B. Sound (mp3 vs wav, Kompression, etc.) Exkurs digitale Formate, Wandlung Analog, Digital, Wellen-form
Vektor- vs. Rastergrafik	Am Beispiel eingescannter Pixelbilder
Eingabe-Verarbeitung-Ausgabe (EVA-Prinzip)	Am Beispiel Menü → Spiel (Steuerung) → Bedingungen → Win/ Fail
Transparenz	Teile eines Bildes, die durchsichtig sind

Hilfestellungen

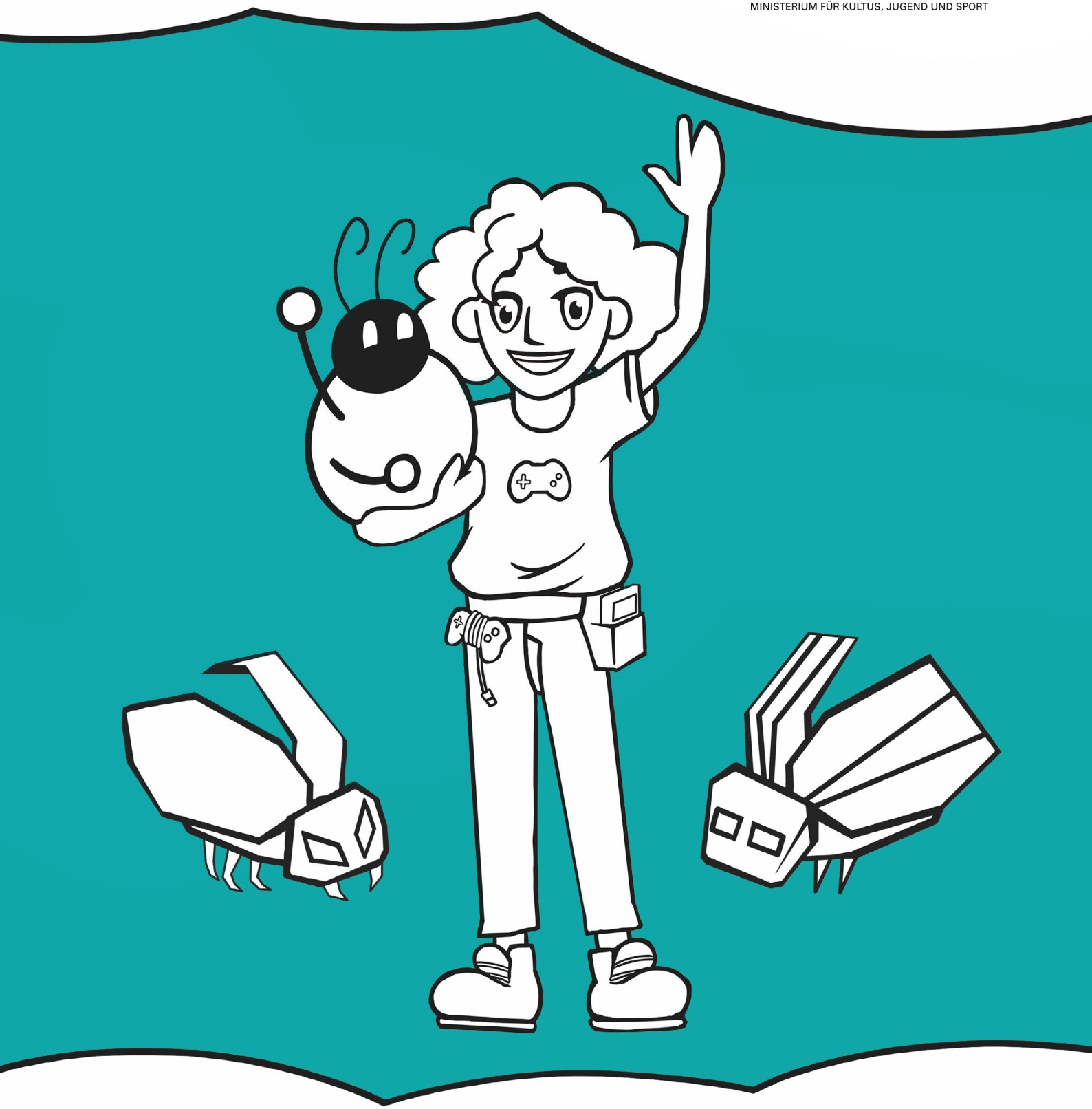
Modul 2 Handbuch (separat)

Ein Projekt der

LFK Die Medienanstalt für Baden-Württemberg

Gefördert durch das

Baden-Württemberg
MINISTERIUM FÜR KULTUS, JUGEND UND SPORT



In Kooperation mit

COMPUTERSPIEL SCHULE STUTTGART spielen entdecken lernen

COMPUTERSPIEL SCHULE FILSTAL spielen entdecken lernen

ZSL Zentrum für Schulqualität und Lehrerbildung Baden-Württemberg